

60.1 XADD—Exchange and Add

Opcode	Instruction	Description
0F C0/r	XADD <i>r/m8,r8</i>	Exchange <i>r8</i> and <i>r/m8</i> ; load sum into <i>r/m8</i> .
0F C1/r	XADD <i>r/m16,r16</i>	Exchange <i>r16</i> and <i>r/m16</i> ; load sum into <i>r/m16</i> .
0F C1/r	XADD <i>r/m32,r32</i>	Exchange <i>r32</i> and <i>r/m32</i> ; load sum into <i>r/m32</i> .

Description

Exchanges the first operand (destination operand) with the second operand (source operand), then loads the sum of the two values into the destination operand. The destination operand can be a register or a memory location; the source operand is a register.

This instruction can be used with a LOCK prefix.

Intel Architecture Compatibility

Intel Architecture processors earlier than the Intel486 processor do not recognize this instruction. If this instruction is used, you should provide an equivalent code sequence that runs on earlier processors.

Operation

```
TEMP ← SRC + DEST
SRC ← DEST
DEST ← TEMP
```

Flags Affected

The CF, PF, AF, SF, ZF, and OF flags are set according to the result of the addition, which is stored in the destination operand.

Protected Mode Exceptions

#GP(0)	<p>If the destination is located in a nonwritable segment.</p> <p>If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.</p> <p>If the DS, ES, FS, or GS register contains a null segment selector.</p>
#SS(0)	If a memory operand effective address is outside the SS segment limit.
#PF(fault-code)	If a page fault occurs.
#AC(0)	If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3.

Real-Address Mode Exceptions

- #GP If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
- #SS If a memory operand effective address is outside the SS segment limit.

Virtual-8086 Mode Exceptions

- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
- #SS(0) If a memory operand effective address is outside the SS segment limit.
- #PF(fault-code) If a page fault occurs.
- #AC(0) If alignment checking is enabled and an unaligned memory reference is made.

60.2 XCHG—Exchange Register/Memory with Register

Opcode	Instruction	Description
90+rw	XCHG AX,r16	Exchange r16 with AX
90+rw	XCHG r16,AX	Exchange AX with r16
90+rd	XCHG EAX,r32	Exchange r32 with EAX
90+rd	XCHG r32,EAX	Exchange EAX with r32
86 /r	XCHG r/m8,r8	Exchange r8 (byte register) with byte from r/m8
86 /r	XCHG r8,r/m8	Exchange byte from r/m8 with r8 (byte register)
87 /r	XCHG r/m16,r16	Exchange r16 with word from r/m16
87 /r	XCHG r16,r/m16	Exchange word from r/m16 with r16
87 /r	XCHG r/m32,r32	Exchange r32 with doubleword from r/m32
87 /r	XCHG r32,r/m32	Exchange doubleword from r/m32 with r32

Description

Exchanges the contents of the destination (first) and source (second) operands. The operands can be two general-purpose registers or a register and a memory location. If a memory operand is referenced, the processor's locking protocol is automatically implemented for the duration of the exchange operation, regardless of the presence or absence of the LOCK prefix or of the value of the IOPL. (See the LOCK prefix description in this chapter for more information on the locking protocol.)

This instruction is useful for implementing semaphores or similar data structures for process synchronization. (See “Bus Locking” in Chapter 7 of the *Intel Architecture Software Developer's Manual, Volume 3*, for more information on bus locking.)

The XCHG instruction can also be used instead of the BSWAP instruction for 16-bit operands.

Operation

```
TEMP ← DEST
DEST ← SRC
SRC ← TEMP
```

Flags Affected

None.

Protected Mode Exceptions

#GP(0)	<p>If either operand is in a nonwritable segment.</p> <p>If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.</p> <p>If the DS, ES, FS, or GS register contains a null segment selector.</p>
#SS(0)	If a memory operand effective address is outside the SS segment limit.
#PF(fault-code)	If a page fault occurs.
#AC(0)	If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3.

Real-Address Mode Exceptions

#GP	If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
#SS	If a memory operand effective address is outside the SS segment limit.

Virtual-8086 Mode Exceptions

#GP(0)	If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
#SS(0)	If a memory operand effective address is outside the SS segment limit.
#PF(fault-code)	If a page fault occurs.
#AC(0)	If alignment checking is enabled and an unaligned memory reference is made.

60.3 XLAT/XLATB—Table Look-up Translation

Opcode	Instruction	Description
D7	XLAT <i>m8</i>	Set AL to memory byte DS:[(E)BX + unsigned AL]
D7	XLATB	Set AL to memory byte DS:[(E)BX + unsigned AL]

Description

Locates a byte entry in a table in memory, using the contents of the AL register as a table index, then copies the contents of the table entry back into the AL register. The index in the AL register is treated as an unsigned integer. The XLAT and XLATB instructions get the base address of the table in memory from either the DS:EBX or the DS:BX registers (depending on the address-size attribute of the instruction, 32 or 16, respectively). (The DS segment may be overridden with a segment override prefix.)

At the assembly-code level, two forms of this instruction are allowed: the “explicit-operand” form and the “no-operand” form. The explicit-operand form (specified with the XLAT mnemonic) allows the base address of the table to be specified explicitly with a symbol. This explicit-operands form is provided to allow documentation; however, note that the documentation provided by this form can be misleading. That is, the symbol does not have to specify the correct base address. The base address is always specified by the DS:(E)BX registers, which must be loaded correctly before the XLAT instruction is executed.

The no-operands form (XLATB) provides a “short form” of the XLAT instructions. Here also the processor assumes that the DS:(E)BX registers contain the base address of the table.

Operation

```
IF AddressSize = 16
    THEN
        AL ← (DS:BX + ZeroExtend(AL))
    ELSE (* AddressSize = 32 *)
        AL ← (DS:EBX + ZeroExtend(AL));
FI;
```

Flags Affected

None.

Protected Mode Exceptions

- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
- If the DS, ES, FS, or GS register contains a null segment selector.
- #SS(0) If a memory operand effective address is outside the SS segment limit.
- #PF(fault-code) If a page fault occurs.

Real-Address Mode Exceptions

- #GP If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
- #SS If a memory operand effective address is outside the SS segment limit.

Virtual-8086 Mode Exceptions

- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
- #SS(0) If a memory operand effective address is outside the SS segment limit.
- #PF(fault-code) If a page fault occurs.

60.4 XOR—Logical Exclusive OR

Opcode	Instruction	Description
34 <i>ib</i>	XOR AL, <i>imm8</i>	AL XOR <i>imm8</i>
35 <i>iw</i>	XOR AX, <i>imm16</i>	AX XOR <i>imm16</i>
35 <i>id</i>	XOR EAX, <i>imm32</i>	EAX XOR <i>imm32</i>
80 /6 <i>ib</i>	XOR <i>r/m8</i> , <i>imm8</i>	<i>r/m8</i> XOR <i>imm8</i>
81 /6 <i>iw</i>	XOR <i>r/m16</i> , <i>imm16</i>	<i>r/m16</i> XOR <i>imm16</i>
81 /6 <i>id</i>	XOR <i>r/m32</i> , <i>imm32</i>	<i>r/m32</i> XOR <i>imm32</i>
83 /6 <i>ib</i>	XOR <i>r/m16</i> , <i>imm8</i>	<i>r/m16</i> XOR <i>imm8</i> (sign-extended)
83 /6 <i>ib</i>	XOR <i>r/m32</i> , <i>imm8</i>	<i>r/m32</i> XOR <i>imm8</i> (sign-extended)
30 /r	XOR <i>r/m8</i> , <i>r8</i>	<i>r/m8</i> XOR <i>r8</i>
31 /r	XOR <i>r/m16</i> , <i>r16</i>	<i>r/m16</i> XOR <i>r16</i>
31 /r	XOR <i>r/m32</i> , <i>r32</i>	<i>r/m32</i> XOR <i>r32</i>
32 /r	XOR <i>r8</i> , <i>r/m8</i>	<i>r8</i> XOR <i>r/m8</i>
33 /r	XOR <i>r16</i> , <i>r/m16</i>	<i>r8</i> XOR <i>r/m8</i>
33 /r	XOR <i>r32</i> , <i>r/m32</i>	<i>r8</i> XOR <i>r/m8</i>

Description

Performs a bitwise exclusive OR (XOR) operation on the destination (first) and source (second) operands and stores the result in the destination operand location. The source operand can be an immediate, a register, or a memory location; the destination operand can be a register or a memory location. (However, two memory operands cannot be used in one instruction.) Each bit of the result is 1 if the corresponding bits of the operands are different; each bit is 0 if the corresponding bits are the same.

Operation

$DEST \leftarrow DEST \text{ XOR } SRC;$

Flags Affected

The OF and CF flags are cleared; the SF, ZF, and PF flags are set according to the result. The state of the AF flag is undefined.

Protected Mode Exceptions

#GP(0)	<p>If the destination operand points to a nonwritable segment.</p> <p>If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.</p> <p>If the DS, ES, FS, or GS register contains a null segment selector.</p>
#SS(0)	If a memory operand effective address is outside the SS segment limit.
#PF(fault-code)	If a page fault occurs.
#AC(0)	If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3.

Real-Address Mode Exceptions

- | | |
|-----|---|
| #GP | If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. |
| #SS | If a memory operand effective address is outside the SS segment limit. |

Virtual-8086 Mode Exceptions

- | | |
|-----------------|---|
| #GP(0) | If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. |
| #SS(0) | If a memory operand effective address is outside the SS segment limit. |
| #PF(fault-code) | If a page fault occurs. |
| #AC(0) | If alignment checking is enabled and an unaligned memory reference is made. |